6. Mahesh

Program Name: Mahesh.java Input File: mahesh.dat

Mahesh is taking an introductory signal processing class for dual credit and has stumbled upon the Hamming Character Code.

The Hamming Character Code is a code in which single-bit errors can be detected and corrected. Each 7-bit ASCII character is embedded in an 11-bit code word called a Hamming character. Bit positions of the hamming character are numbered from most significant to least significant. The encoding is done as follows:

Bit positions:	1	2	3	4	5	6	7	8	9	10	11
Bits:	p	p	d	p	d	d	d	p	d	d	d

Bits 1, 2, 4, and 8, all having bit positions that are a power of two, are **parity bits**. Bits 3, 5, 6, 7, 9, 10, and 11 are the **data bits** which represent the most to least significant bits of the embedded ASCII character. The parity bits are computed so that

bit 1 = (bit 3 + bit 5 + bit 7 + bit 9 + bit 11) mod 2
bit 2 = (bit 3 + bit 6 + bit 7 + bit 10 + bit 11) mod 2
bit 4 = (bit 5 + bit 6 + bit 7) mod 2
bit 8 = (bit 9 + bit 10 + bit 11) mod 2

For example, the ASCII character ‘M’ having 7-bit code 1001101 will be represented by 01110010101.

Suppose the text encoded as a sequence of Hamming characters is transmitted through a channel that can introduce at most one bit error per Hamming character. We may determine the location of the error and correct it as follows. If the Hamming character 01110010101 representing ‘M’ were to be received as 01110010001, the location of the errant bit may be computed as the sum of the positions of the parity bits that disagree with their recomputed values. The offending bit is then complemented to determine the correct Hamming character. Thus the received word 01110010001

∗ bit 1 is 0	1 = (bit 3 + bit 5 + bit 7 + bit 9 + bit 11) mod 2
bit 2 is 1	1 = (bit 3 + bit 6 + bit 7 + bit 10 + bit 11) mod 2
bit 4 is 1	1 = (bit 5 + bit 6 + bit 7) mod 2
∗ bit 8 is 0	1 = (bit 9 + bit 10 + bit 11) mod 2

1 + 8 = 9, which indicates the location of the offending bit! If bit 9 in 01110010001 is changed, it yields the corrected Hamming character for ‘M’, 01110010101. This decoding method works for all Hamming characters with one bit error. Naturally, if there are no bit errors, there will be no discrepancies between parity bits and their recomputed values.

Can you help Mahesh write a program to implement the bit correction described above?

Input: Input starts with a line containing an integer N (1<=N<=150), the number of test cases. Each of the following N lines contains an integer, representing a single Hamming character with at most one bit error. Write a program that decodes the message, correcting for single-bit errors.

Output: The decoded message should be written to the standard output as characters, not integers, all on the same line with no separation.

Sample Input and Output on next page

Mahesh sample input:

```
7
22992
3533
-20667
24407
14937
-17578
23535
```

Sample Output:
```
Hamming
```

Explanation of Sample Output

Input Integer	Input Integer in Hexadecimal	Low-Order 11 Bits	Offending Bit	Resulting Character
22992	59D0	00111010000	5	1001000 = H
3533	0DCD	10111001101	9	1100001 = a
-20667	AF45	11101000101	7	1101101 = m
24407	5F57	11101010111	10	1101101 = m
14937	3A59	01001011001	3	1101001 = i
-17578	BB56	01101010110	0	1101110 = n
23535	5BEF	01111101111	6	1100111 = g